Machine Vision and Applications (1989) 2:81–102

Machine Vision and Applications

Range Estimation from Intensity Gradient Analysis

Kurt Skifstad and Ramesh Jain

Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, Michigan 48109-2122, USA

Abstract: Conventional approaches to recovering depth from gray-level imagery have involved obtaining two or more images, applying an "interest" operator, and solving the correspondence problem. Unfortunately, the computational complexity involved in feature extraction and solving the correspondence problem makes existing techniques unattractive for many real-world robotic applications. By approaching the problem from more of an engineering perspective, we have developed a new depth recovery technique that *completely avoids* the computationally intensive steps of feature selection and correspondence required by conventional approaches. The Intensity Gradient Analysis technique (IGA) is a depth recovery algorithm that exploits the properties of the MCSO (moving camera, stationary objects) scenario. Depth values are obtained by analyzing temporal intensity gradients arising from the optic flow field induced by known camera motion. In doing so, IGA avoids the feature extraction and correspondence steps of conventional approaches and is therefore very fast. A detailed description of the algorithm is provided along with experimental results from complex laboratory scenes.

Key Words: depth recovery, intensity gradient, motion stereo, optic flow, stereopsis

1 Introduction

The problem of depth recovery has received a great deal of attention by vision researchers (Nevatia 1976; Grimson 1981; Nishihara 1984; Prazdny 1985; Herman and Kanade 1986; Yachida et al. 1986; Bolles et al. 1987; Clement 1987; Jain et al. 1987; Tsukiyama and Huang 1987; Xu et al. 1987; Boyer and Kak 1988; Griswold and Yeh 1988). Although conceptually appealing and theoretically elegant, the existing techniques are, as a whole, usually computationally expensive. This computational burden makes most of these techniques unacceptable for real-world robotic applications. By approaching the problem from more of an engineering perspective, we have developed a new depth recovery technique that *completely avoids* the computationally intensive steps of feature selection and correspondence required by conventional approaches.

The Intensity Gradient Analysis technique (IGA) was born largely out of frustration arising from both practical and philosophical considerations. Practically, existing techniques are inappropriate for many real-world applications due to their computationally intensive nature. Philosophically, it seems that the depth recovery problem in computer vision has been motivated primarily by a desire to emulate human (or at least binocular) vision. Too little attention has been given to the purely "robotic" perspective; that is, it seems that more people have spent their efforts "trying to implement their vision algorithms on computers" than trying to "make machines see." Certainly, it is easy to formulate the depth recovery from a binocular perspective: take two images, find matching points, compute depths. However, an often ignored fact is that this "extract and match" paradigm is required *only* when the images have been acquired from two quite disparate points. This is not necessarily bad, but it is also not necessarily the best path to take if one is trying to achieve the illusive nirvana of practical, efficient vision algorithms that *actually work.*

By approaching the depth recovery problem from the classic "engineering"[1] perspective, the IGA algorithm was developed. Essentially, the IGA algorithm arose from one simple observation:

Address reprint requests to: K. Skifstad, Artificial Intelligence Laboratory, Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, Michigan 48109-2122, USA.

[1] "This is the problem I have to solve, this is environment I am to solve the problem in, and these are the tools I have. Now, how can I solve the problem?"

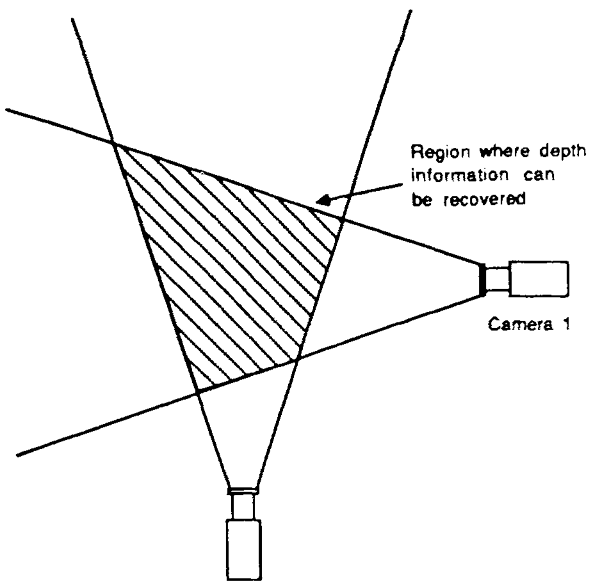

Figure 1. Range Information can only be recovered for objects in the field of view of both sensors (shaded area).

- Camera motion[2] dictates object motion: Because of the optical flow field induced by camera motion, stationary objects displace (in the image) with respect to the observer along vectors passing through the focus of expansion (FOE). How these objects displace depends on their proximity to the camera and the direction of camera motion.

Given this observation and the proper insight, it turns out that the depth recovery problem is quite straightforward.

This article begins with a brief review of existing techniques and the general problem of depth recovery. The theoretical basis for the IGA algorithm is discussed. Experimental results on complex laboratory sequences are given and future extensions of the IGA algorithm are discussed.

2 Computational Stereo

Fundamentally, the depth recovery problem (as perceived by most researchers) is described by the five-step process specified by Barnard and Fischler (1982):

1. Image acquisition
2. Camera modeling
3. Feature extraction
4. Image matching
5. Depth determination

[2] Unless specified otherwise, all references to camera motion will refer to translational motion; that is, that in which the camera displaces along a linear path with fixed orientation.

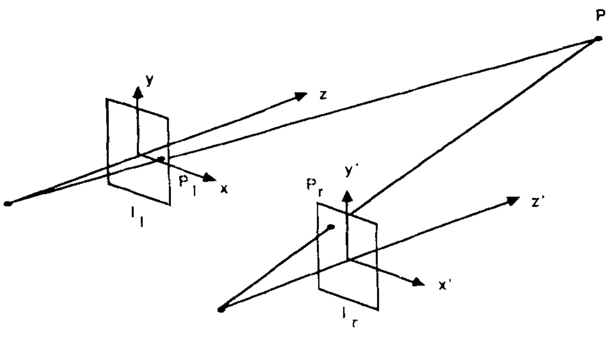

Figure 2. The Binocular Stereo Camera model.

As image acquisition involves the well-known process of receiving information from the sensing device, and depth determination is a straightforward task once image matching is done, it is not surprising that the bulk of the research efforts in this area have focused on camera modeling, feature extraction, and image matching.

2.1 Camera Modeling

The camera model defines the imaging geometry for the system. Since depth values can only be recovered for objects that are contained in both images, the imaging geometry defines the region in space where information can be extracted (see Figure 1). In addition, the way the camera model is set up is often used to constrain the search for matching points in the images. For example, consider two of the more popular imaging geometries: the Binocular Stereo model (Figure 2), and the Axial Motion Stereo model (Figure 3).

The binocular stereo model has been the choice of most researchers (Grimson 1981; Barnard and Fischler 1982; Clement 1987; Boyer and Kak 1988; Griswold and Yeh 1988) and certainly is the choice for researchers whose purpose is to study human vision (Grimson 1981; Mayhew and Frisby 1981). In the binocular, or left–right model, the cameras may be aligned so that corresponding scanlines in the two images lie in the same epipolar plane. This simplifies the solution of the correspondence problem by limiting the search space to a single dimension.

The axial motion stereo model (O'Brien and Jain 1984) is often used to exploit known camera motion (Itoh et al. 1981; Jain et al. 1987). Animals are known to use a similar technique, called looming, to locate objects (prey, food, etc.) (Collet and Harkness 1982). In the axial motion stereo paradigm, the disparity between matching points (and therefore ambiguity) is potentially greater the farther a point is from the focus of expansion (FOE). Alvertos et al. (1988) have shown that by first matching points close to the FOE, fewer ambiguous and false

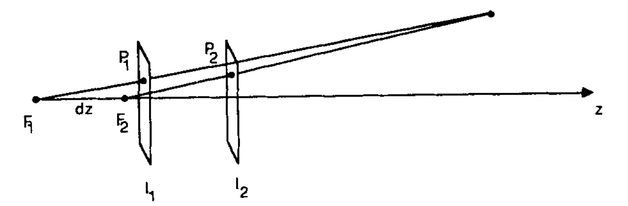

Figure 3. The Axial Motion Stereo camera model.

matches occur than in other (conventional) stereo paradigms.

2.2 Image Matching: The Correspondence Problem

In order to recover depth at a given point in the image (using the conventional approach), the corresponding point must be found in the other of the image pair. Consider the situation shown in Figure 4. The first image (left) shows a set of points in the field of view. We may then be able to infer, for example, that point A in the first image corresponds to point Q in image 2, point D to point S, point E to point T, and so on. However, a problem arises when one tries to find a match for points B and C. Does point B correspond to point R? Does point C correspond to point R? Do they both perhaps correspond to point R? Or, neither B nor C corresponds to R?

Not surprisingly, a great deal of effort has been focused on simplifying and speeding up the solution of the correspondence problem (Nishihara 1984; Yachida et al. 1986; Bolles et al. 1987; Xu et al. 1987; Baker and Bolles 1988). To find the correct correspondences in complicated scenarios, some assumptions about spatial continuity must be made and then some sort of relaxation algorithm can be employed to find the correct matches (Barnard and Thompson 1980). Often this technique is applied at several layers of resolution, with the matches found at coarser resolution used to guide the matching process at finer resolutions (Grimson 1981). Small camera movements can also be used to reduce the possible disparities, thus constraining the area searched to find correspondences (Bolles et al. 1987; Xu et al. 1987).

2.3 Feature Extraction

Even assuming that the correspondence problem can be solved, it would be both impractical and improbable to find a match for each and every point in an image. Consider, for example, the complexity of finding 65,536 out of 65,536 correct matches in a 256 × 256 image, or correctly matching points in a region of constant intensity. Therefore, before correspondences can be determined, a subset of the im-

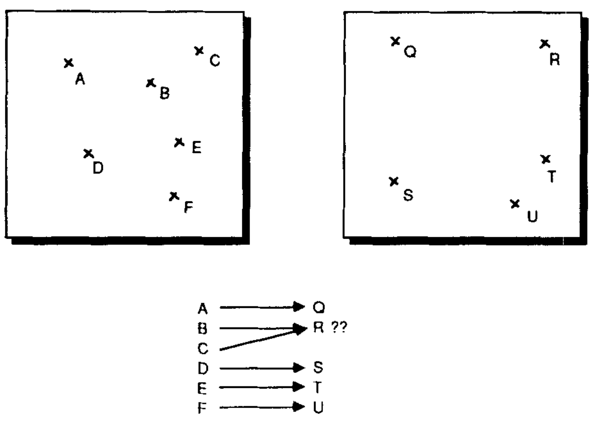

Figure 4. The Correspondence Problem: Which point corresponds to R?

age must be selected for matching. Certainly, this subset should be relatively sparse (to facilitate matching), but still capture the "essence" of the image. Often, this subset consists of features such as:

- Zero crossings of the Laplacian of the Gaussian (Grimson 1981; Nishihara, 1984)
- Corners (Moravec 1981; Jain et al. 1987)
- Edges (Jain 1984; Tsukiyama and Huang 1987)

Issues involved in selecting the type of feature to look for include operator size, robustness, and localization. If speed is a major consideration, the size of the operator needs to be kept to a minimum because a large operator increases the computational burden on the system. Larger operators, however, because they have more information available, tend to be more robust, giving fewer false responses. On the other hand, smaller operators provide better localization, making results more accurate.

2.4 Computational Complexity

Combining the problems of feature extraction and correspondence, it is not difficult to see that existing techniques are quite computationally intensive. Consider, for example, Grimson's (1981) implementation of the Marr-Poggio algorithm. To determine depth values one must:

- Find the zero-crossings of the Laplacian of the Gaussian of the two images *at four levels of resolution.*
- For each of the four image pairs, find correspondence using the information at lower resolutions to guide the search.

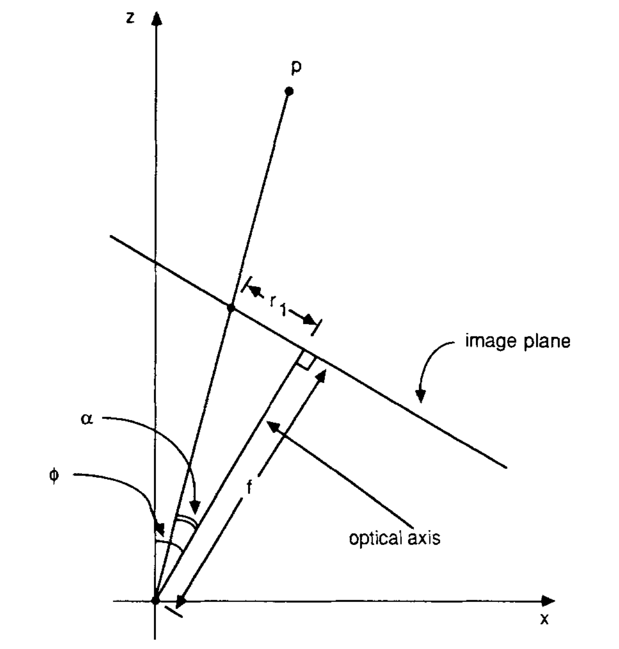

Figure 5. **Camera with focal length f located at the origin of an arbitrary $x - z$ coordinate system. The camera's optical axis makes an angle ϕ with respect to the z-axis.**

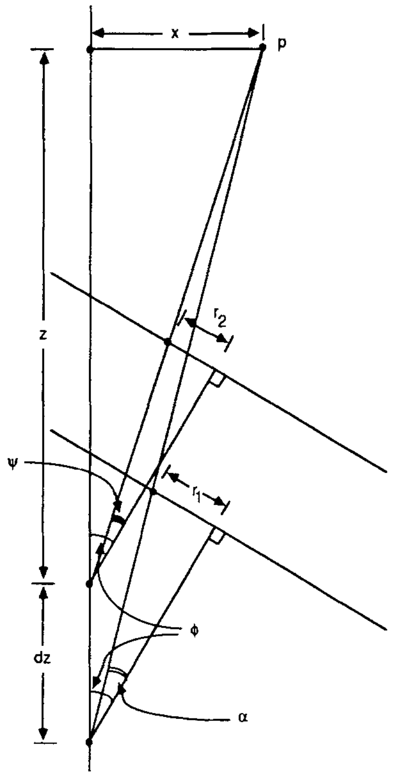

Figure 6. **The camera is displaced a distance dz along the z-axis. Depth can be recovered using information recovered from the two images, and information regarding camera orientation.**

- From the correspondences found at the finest resolution, create a disparity map.
- Using knowledge of the imaging geometry and the disparity map, determine depth values.

Considering the computational effort involved in the feature extraction and finding correspondences, it certainly would be nice to be able to avoid (or at least simplify) these steps.

3 The Intensity Gradient Analysis Technique (IGA)

Researchers have shown that the analysis of temporal intensity gradients is a useful tool for recovering camera position and orientation (Lucas 1985; Lucas and Kanade 1985; Negahdaripour and Horn 1986, 1987). The Intensity Gradient Analysis technique (IGA) also uses temporal intensity gradients, but purely on a local level. IGA uses a moving camera to induce an optic flow field on the image, causing stationary objects in the field of view to displace in known directions (Prazdny 1980). This displacement results in temporal intensity gradients at locations in the images through which the objects pass. Because these temporal gradients are entirely dependent on the proximity of the objects in the field of view, these gradients turn out to be reliable cues for recovering depth.

This new "IGA paradigm" requires *far less computational effort* than the conventional approach, and completely avoids the feature extraction and correspondence steps of the conventional approach.

The description of the IGA algorithm is prefaced by a short introduction to computing depth using motion stereo. This is used to motivate the need to compute disparity. It should be noted, however, that this does not imply that the IGA technique is limited to these two camera models. IGA can be applied to image sequences obtained using *arbitrary* translational camera motion, as long as that motion is properly controlled.

3.1 Computing Depth Using Motion Stereo

Consider a camera of focal length f placed at the origin in an arbitrary x–z coordinate system so that the optical axis of the camera makes an angle ϕ with respect to the z axis (Figure 5). The depth of point p cannot be recovered because there is not enough information present to uniquely specify that value. Suppose, then, that the camera is displaced some distance dz along the positive z axis (Figure 6). It is now possible to express the depth z of point p:

$$z = dz \frac{\tan(\phi - \alpha)}{\tan(\phi - \psi) - \tan(\phi - \alpha)} \tag{1}$$

where

$$\alpha = \arctan\left(\frac{r_1}{f}\right) \tag{2}$$

and

$$\psi = \arctan\left(\frac{r_2}{f}\right) \tag{3}$$

where r_1 and r_2 are the distance from the center of the image to the projection of point p in the first and second images, respectively. Equation (1) then becomes:

$$z = dz\,\frac{\cos(\phi - \psi)\sin(\phi - \alpha)}{\sin(\alpha - \psi)} \tag{4}$$

Since f is a known parameter, and r_1 and r_2 are measurable quantities, it is desirable to express this equation in terms of $\tan\alpha$ and $\tan\psi$.

Through trigonometric substitution, equation (4) expands to:

$$z = dz\,\frac{(\cos\phi\cos\psi + \sin\phi\sin\psi)(\sin\phi\cos\alpha - \cos\phi\sin\alpha)}{\sin\alpha\cos\psi - \cos\alpha\sin\psi} \tag{5}$$

which, in turn, becomes:

$$z = dz\,\frac{\sin\phi\cos\phi + \sin^2\phi\tan\psi - \cos^2\phi\tan\alpha - \sin\phi\cos\phi\tan\alpha\tan\psi}{\tan\alpha - \tan\psi} \tag{6}$$

substituting in $\tan\alpha = r_1/f$, $\tan\psi = r_1/f$, and $r_2 = r_1 + \delta$, where δ is disparity, gives us:

$$z = \frac{dz \times f}{-\delta} \times \left\{\sin\phi\cos\phi + \frac{\delta\sin^2\phi}{f} + \frac{r_1}{f}\left[\sin^2\phi - \cos^2\phi - \frac{(r_1 - \delta)}{f}\sin\phi\cos\phi\right]\right\} \tag{7}$$

In some cases (e.g., lateral motion stereo), it may be desirable to compute x, the distance from the axis of motion to the object (Figure 7), where

$$x = z \times \tan(\phi - \psi) \tag{8}$$

Note that in the special cases where $\phi = 0$ (axial motion) and $\phi = \pi/2$ (lateral motion), equations 7 and 8 reduce to the more familiar time to collision ratio (Lee 1976, 1980):

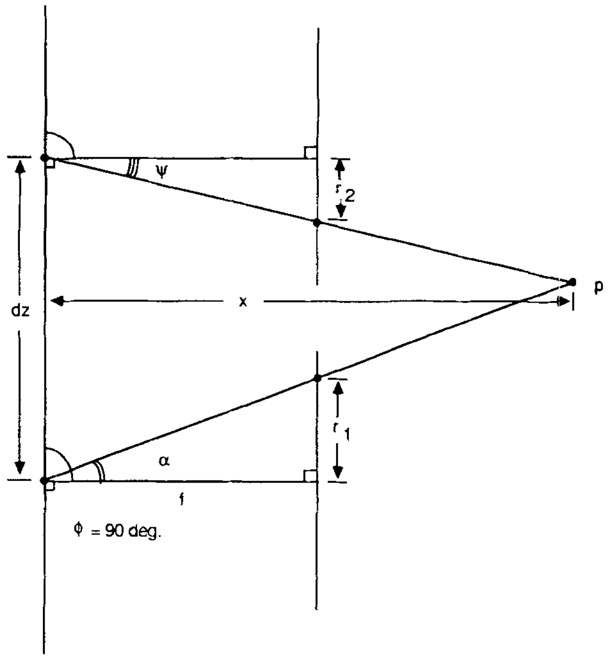

Figure 7. For lateral motion, one may wish to calculate x, the distance from the axis of motion to the object.

$$\frac{z}{dz} = \frac{r_1}{\delta} \tag{9}$$

and the binocular stereo equation (Horn 1986):

$$x = \frac{dz}{\delta}f \tag{10}$$

To determine z (or x), we must find dz, ϕ, f, δ (disparity), and r_1. Because the camera motion is controlled precisely, dz and ϕ are known. Since it is simply a parameter of the lens we are using, f is known, and r_1 can be found by measuring the distance (in pixels) from the object to the center of the image. Unfortunately, the final piece of the puzzle is still missing because it is not, in general, possible to determine δ without explicitly addressing the correspondence problem or determining the optical flow.

3.2 Finding Disparity Without Solving Correspondence

While it may not be possible to recover arbitrary disparity without explicitly addressing the correspondence problem, it turns out that it *is* possible to recover a specific disparity (namely, $\delta = 1$) and, thus, recover depth without solving the correspondence problem. This follows directly from the principles of image formation.

We know the gray-level I recorded at a given pixel is proportional to the number of light quanta

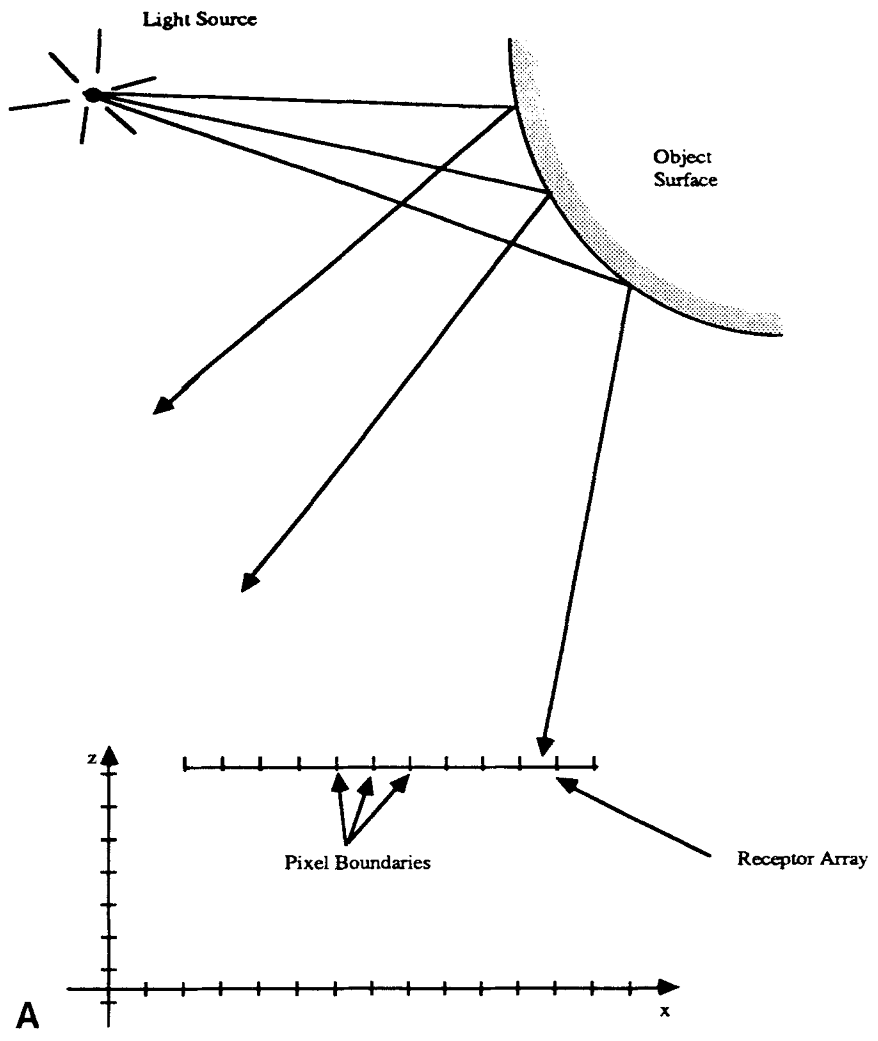

Figure 8. A one-dimensional receptor array translating in an arbitrary coordinate system.

incident on that region in the image. This can be expressed as:

$$I = k \int \int \rho(x,y)\, dxdy \tag{11}$$

With k being the constant of proportionality and $\rho(x,y)$ being the quantum catch at point (x,y).

Given this information, consider the one-dimensional example shown in Figure 8A. Assuming that all pixels have the same spectral sensitivity, we can compute the gray-level recorded at pixel n ($I(n)$) as follows:

$$I(n) = k \int_{nx_p}^{(n+1)x_p} \rho(x)\, dx \tag{12}$$

Where x_p is the width of one pixel.

Suppose, then that the camera is moved such that the translational component of the motion induced[3] on the object is x_t and that the axial component z_a is very small compared to the distance of any objects in the field of view ($z_a \ll z, \forall z$). This is shown in Figure 8B. We can now compute the gray-level recorded at pixel n from our new camera location:

$$I(n) = k \int_{nx_p - x_t}^{(n+1)x_p - x_t} \rho(x)\, dx \tag{13}$$

$$= k \int_{nx_p}^{(n+1)x_p} \rho(x + x_t)\, dx \tag{14}$$

Let us now look at the special case when $x_t = x_p$.

$$I(n) = k \int_{nx_p}^{(n+1)x_p} \rho(x + x_p)\, dx \tag{15}$$

[3] Remember that a moving camera induces an optical flow field on the image, causing stationary objects to displace (in the image) along vectors through the FOE.

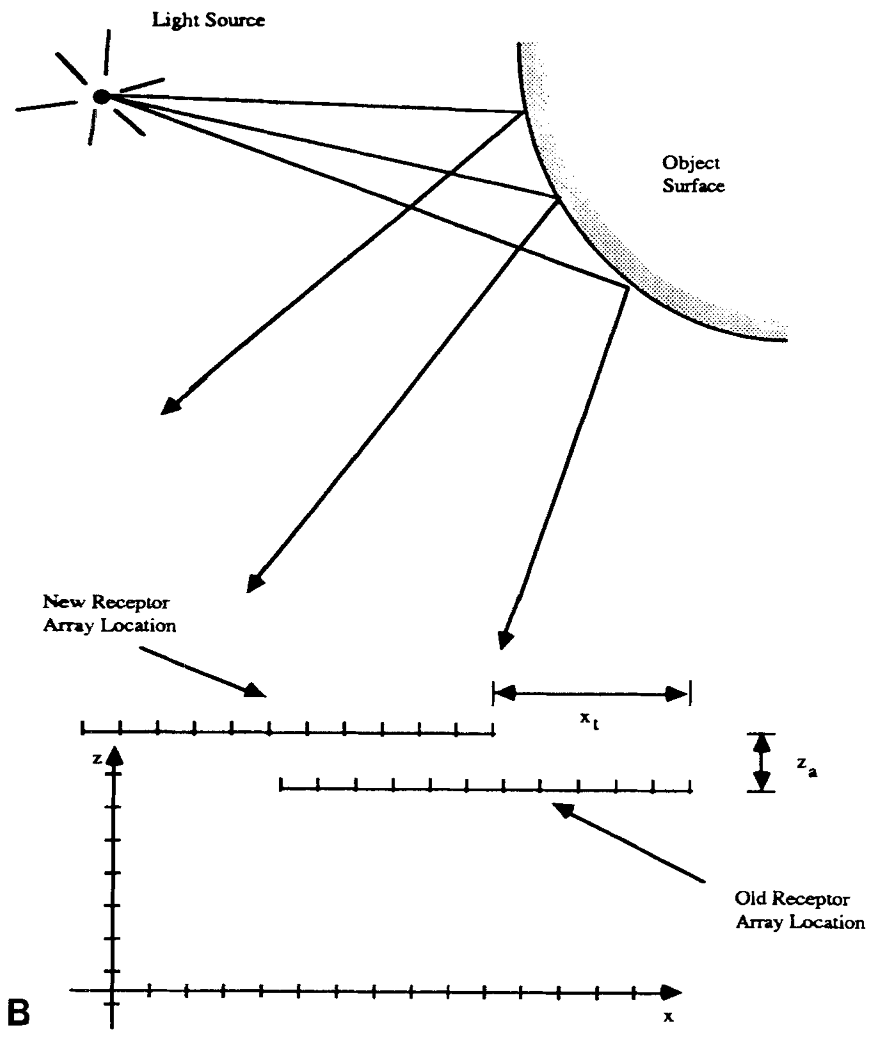

Figure 8. *continued*

$$= k \int_{nx_p - x_p}^{(n+1)x_p - x_p} \rho(x)\, dx \tag{16}$$

$$= k \int_{(n-1)x_p}^{(n)x_p} \rho(x)\, dx \tag{17}$$

$$I(n) = I_{prev}(n - 1) \tag{18}$$

Where $I_{prev}(n - 1)$ is the intensity recorded at location $(n - 1)$ before the camera was moved.

What this tells us, not surprisingly, is that if an object displaces one pixel, the intensity perceived at the location the object moved *into* must equal the intensity perceived before the displacement took place at the location the object moved *out of*. Of course, for this displacement to be detected, it must occur at a point in the image where such an event is perceivable. That is, it must occur at a point in the image where the *spatial intensity gradient* (along the induced displacement vector) is nonzero. Points in the interior of regions of constant intensity (spatial gradient equals zero) provide no depth cues (Goldstein 1984) as there is no way of uniquely assigning image points objects (consider the phenomenon of snow-blindness).

So, following the above line of reasoning, if a full pixel displacement is to occur, the *temporal intensity gradient* (change in intensity between frames) at a location in the images must equal the *spatial intensity gradient* (change along the displacement vector) at that same location in the first image:

$$\frac{\delta I(x,t_1)}{\delta t} = \frac{\delta I(x,0)}{\delta x} \tag{19}$$

or

$$I_1(n) - I_0(n) = I_0(n) - I_0(n - 1) \tag{20}$$

So, the problem of recovering disparity (and therefore depth) is reduced to monitoring temporal

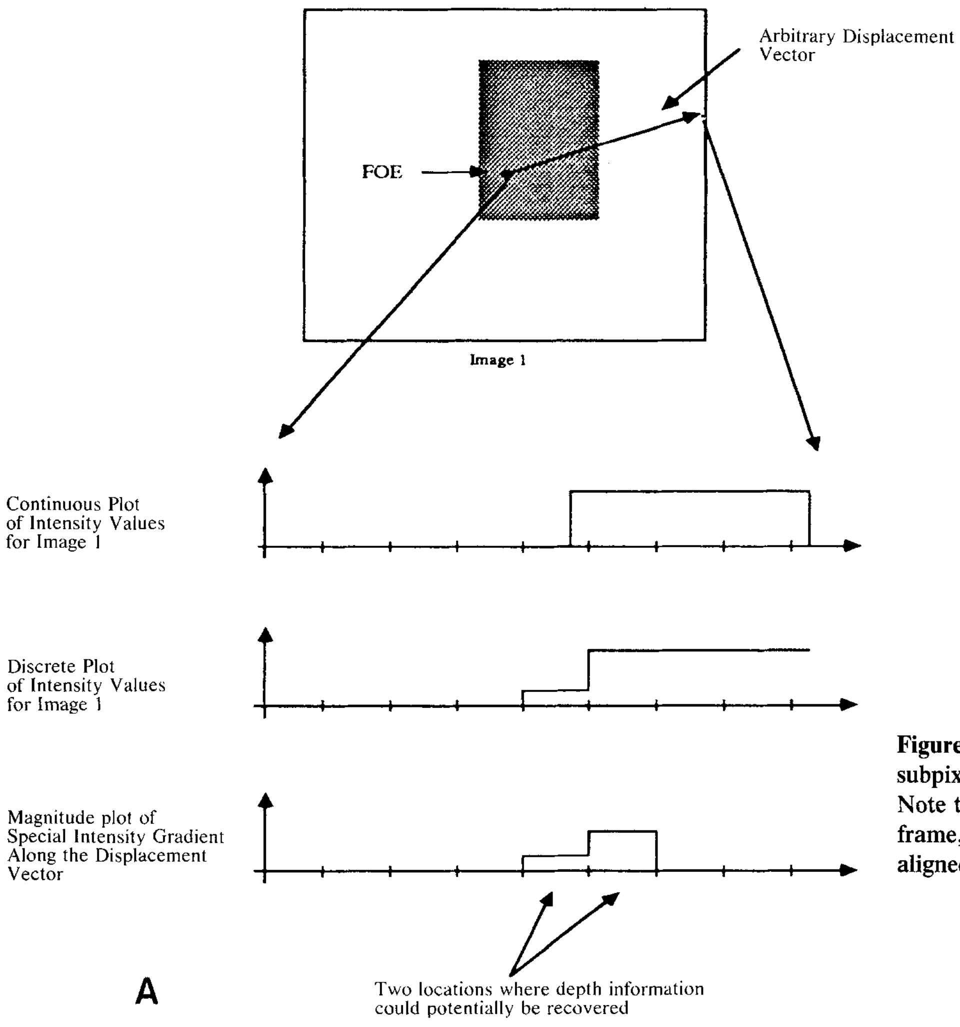
Arbitrary Displacement Vector
FOE
Image 1
Continuous Plot of Intensity Values for Image 1
Discrete Plot of Intensity Values for Image 1
Magnitude plot of Special Intensity Gradient Along the Displacement Vector
A
Two locations where depth information could potentially be recovered

Figure 9. An illustration of the subpixel displacement problem. Note that in (A), the reference frame, the object boundary is *not* aligned with the pixel boundary.

intensity gradients. When the temporal gradient equals the spatial gradient, disparity is one, and depth can be recovered using the following equation:

$$z = -dz \times f \times \left\{ \sin\phi\cos\phi + \frac{\sin^2\phi}{f} + \frac{r_1}{f}\left[\sin^2\phi - \cos^2\phi - \frac{(r_1 - 1)}{f}\sin\phi\cos\phi\right]\right\} \tag{21}$$

which, for axial and lateral motion, reduces to:

$$z = dz \times r_1 \tag{22}$$

and

$$x = dz \times f \tag{23}$$

respectively.

3.3 The Subpixel Displacement Problem

Unfortunately, the assumption that an object must displace a distance of one pixel or more before the temporal intensity gradient equals the spatial intensity gradient does not hold in every case. Consider Figure 9. Figure 9A shows an image and the corresponding intensity plots (continuous and discrete) along an arbitrary displacement vector through the FOE. Since this scene shows a uniformly shaded object against a uniform background, there is only one place where we can find depth cues: at the object boundary. Note that, in this case, the object boundary constitutes a step edge in the continuous space, but, since the object boundary does not lie on a pixel boundary, a staircase effect is produced in the discrete domain. The magnitude of the discrete spatial intensity gradient along this vector is also given. Figure 9B shows a second image of the same scene, with the observer closer to the object. In this image, the boundary of the object has dis-

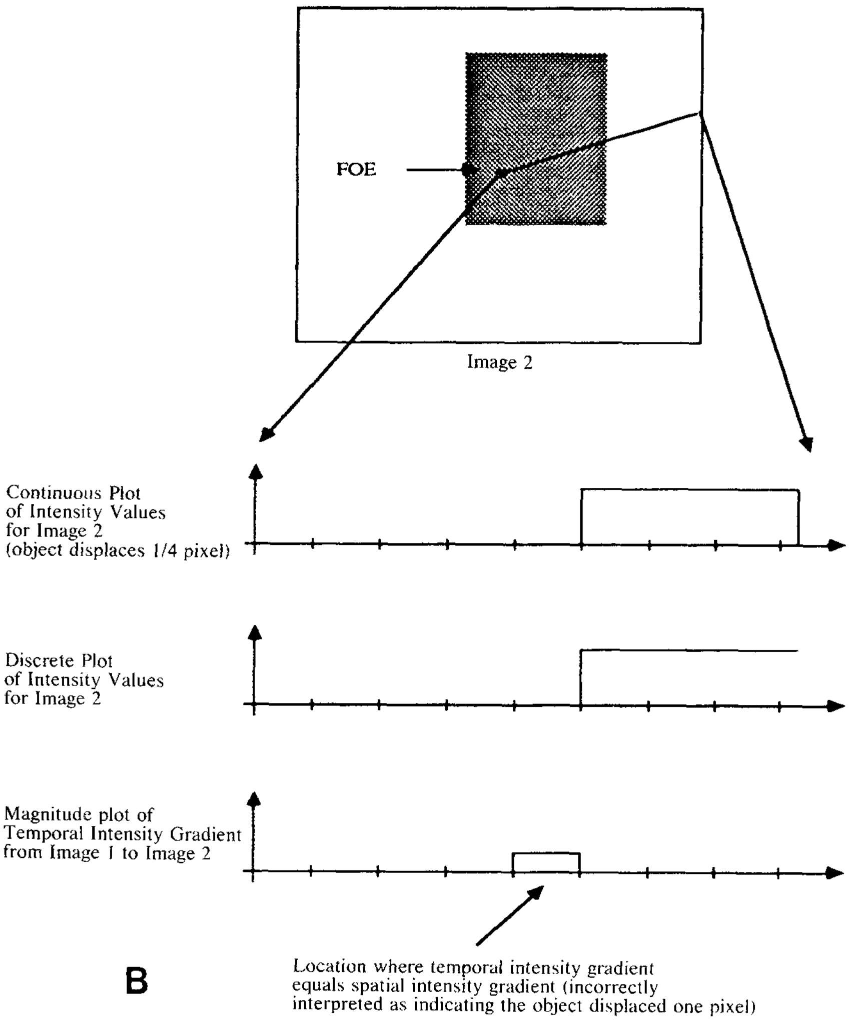
FOE
Image 2
Continuous Plot of Intensity Values for Image 2 (object displaces 1/4 pixel)
Discrete Plot of Intensity Values for Image 2
Magnitude plot of Temporal Intensity Gradient from Image 1 to Image 2
B
Location where temporal intensity gradient equals spatial intensity gradient (incorrectly interpreted as indicating the object displaced one pixel)

Figure 9. *continued*

placed a small distance toward the periphery, causing the intensity discontinuity due to the object boundary to align with a pixel boundary. Intensity plots are shown as well as a magnitude plot of the temporal intensity gradient, as compared to the first image (Figure 9A). Since the object was not lined up on a pixel boundary in the first frame, a displacement of a fraction of a pixel was sufficient for the temporal gradient (between image 1 and image 2) to equal the spatial gradient (in the reference image) at the location in the image where the object boundary originally was located.

Figure 10 shows several characteristic intensity profiles where an object displacement of a fraction of a pixel will induce a temporal intensity gradient equal to the spatial intensity gradient.

3.4 Solving the Subpixel Displacement Problem

Since camera motion is known, we know the direction and path along which an object must displace. Furthermore, we know that depth can only be recovered at those locations in the image where the spatial intensity gradient is nonzero. Let us call those points *interesting* or I-points. And, for the sake of completeness, let us call the remaining points, those belonging to regions of constant intensity, *smooth* or S-points.

Consider what happens as an object displaces across the image: both the I- and S-points corresponding to the object will displace (only I-points will give us cues for depth).

What do we know about the behavior of these points? When an I-point moves into a new position, the transition is well-defined. We know that the intensity profile of the object at the I-point in the reference image (used to compute the spatial intensity gradient), so we know how the perceived intensity should change at the point the I-point moves into. But, what about the location that the I-point moves out of?

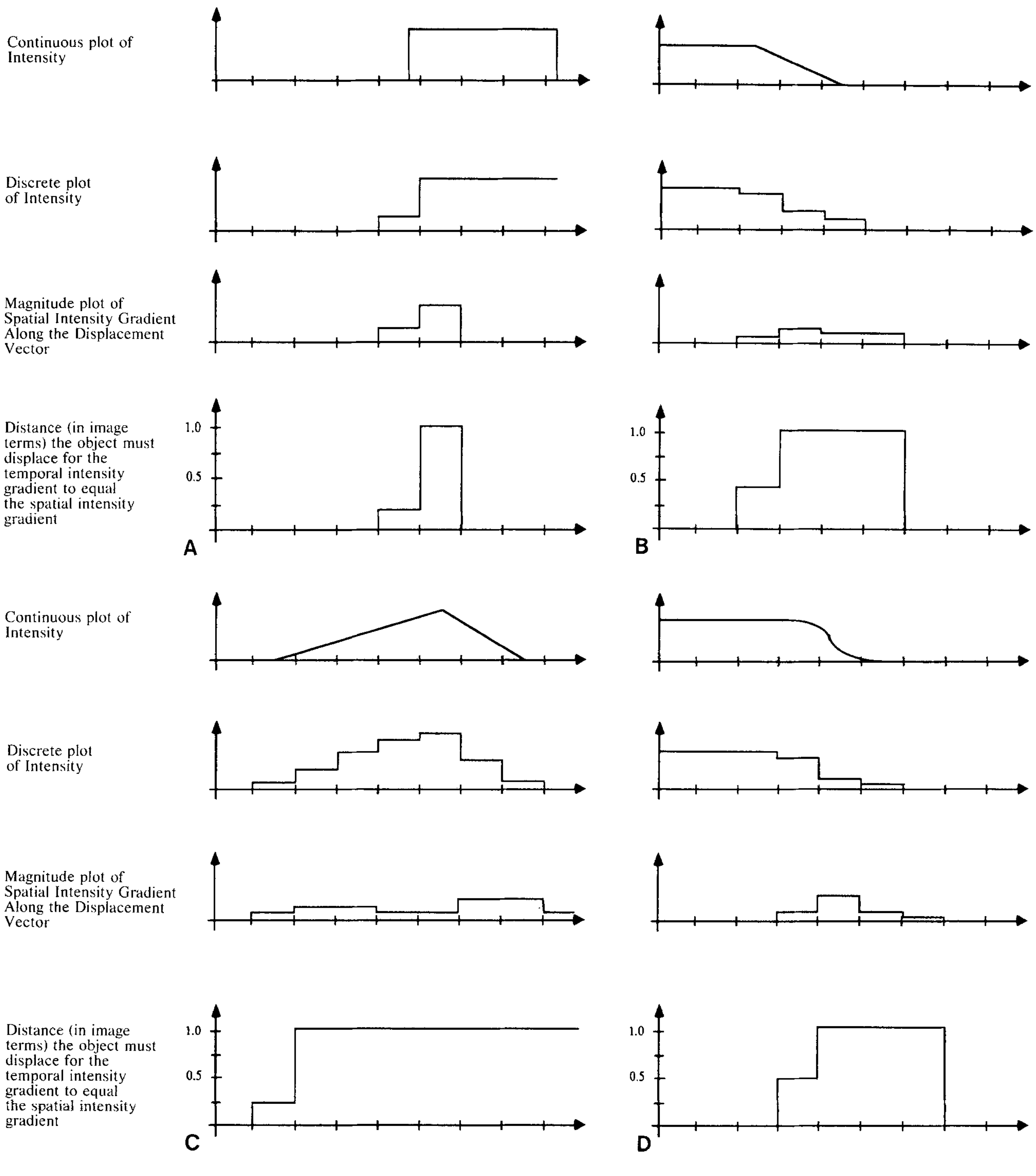

Figure 10. Several examples where an image displacement of a fraction of a pixel will cause the temporal intensity gradient to equal the spatial intensity gradient, incorrectly indicating a full pixel displacement: (A) step edge; (B) ramp edge; (C) peak; (D) slope.

When the image location vacated by an *I*-point is filled by another *I*-point, we have no problem because another *I*-point is moving into the vacated spot (and the transition is, therefore, well-defined). But, when an *S*-point moves into this location, there is potential for error. Because we are working in a discrete universe, an *S*-point adjacent to an *I*-point *does not imply* that the object surface ceases to have uniform shading at the pixel boundary (Figure 10A–D). So, assuming that the actual shading discontinuity lies somewhere inside the region corresponding to the *I*-point, the object need only displace a fraction of a pixel for the temporal intensity derivative to equal the spatial intensity derivative at this location. This will lead to the incorrect assumption that the object displaced a distance equal to a full pixel, resulting in the calculated depth being less than the true value.

Since there is no way of detecting when the *I–S* boundary corresponds to the actual shading discontinuity, we must avoid those situations that will potentially lead to error. Fortunately, not all *I–S* boundary points need be ignored. In fact, only half need be rejected from consideration. This follows from the fact that half of the *I–S* boundary positions will have *I*-points moving into locations vacated by *S*-points, which, as described above, is a well-defined transition and can be used to accurately compute depth.

How do we know which *I–S* boundaries are *I* to *S* transitions and which are *S* to *I* transitions? This is where our knowledge of camera motion comes into play. Since we know the camera motion parameters, we know where the FOE is. Since we know where the FOE is, we know what direction objects must displace. And, since we know what direction objects must displace, we know which *I–S* boundaries are potential *I* to *S* transitions and which *I–S* boundaries are potential *S* to *I* transitions. Therefore, we know exactly where the subpixel displacement problem may occur, and can ignore those points.

3.5 Extending to Two Dimensions

Since the previous discussion has been in terms of one-dimensional images, one might think that the technique needs to be extended to deal with conventional two-dimensional imagery. Such an extension is not necessary, however, since, in this scenario, depth recovery is purely a one-dimensional problem. Consider Figure 11. Since we know camera motion, we know the FOE. Therefore, we also know the trajectories the objects must follow. To determine depth at a particular point, we need only know information in the neighborhood of the point in question along its (one-dimensional) displacement vector.

An analogy can be made between the flow (displacement) vectors from motion stereo and epipolar lines in conventional stereo. In both cases, researchers have used their properties to reduce depth recovery to a one-dimensional problem (Bolles et al. 1987; Jain et al. 1987).

3.6 The IGA Algorithm

Figure 12 shows the steps for the IGA algorithm. As can be seen, this technique is very straightforward,

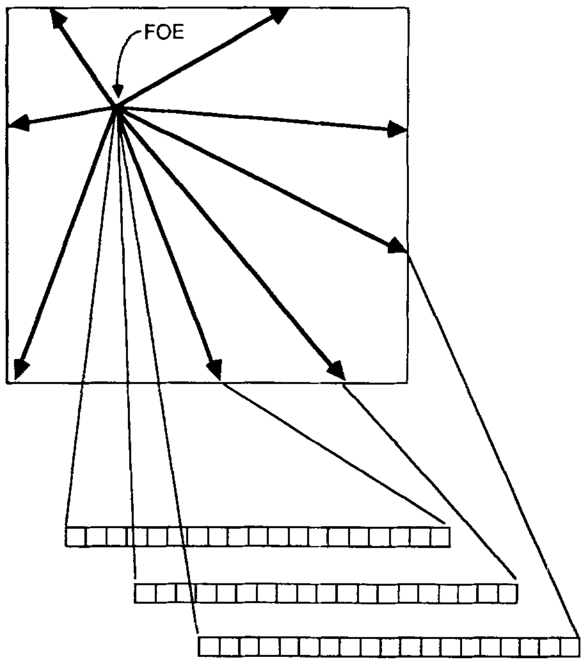

Figure 11. Applying IGA to two-dimensional images involves looking at several one-dimensional problems.

and requires minimal computational effort. The initialization phase requires the acquisition of the reference image, and the computation of the spatial intensity gradient at each location in the image.

The Depth Recovery Loop portion of the algorithm requires the camera to be moved, an image to be acquired, and the temporal gradient to be computed. Since we know (from the reference image) where the regions of constant intensity are, we need not compute the temporal gradient at all points in the new image. Only those points corresponding to *I*-points in the reference image need to be considered.

Once the temporal gradient is computed, it is compared to the spatial gradient. If equal (or greater than), the point is considered for depth determination. However, before we can compute depths, we must be sure that this point doesn't correspond to some previously perceived object displacing across the image. Since we know the distance to each object and how far the camera has moved, we can use equation (7) to determine how far across the image an object has displaced since it first was perceived:

$$\delta = dz \times f \times \frac{\sin\phi\cos\phi + \frac{r_1}{f}\left[\sin^2\phi - \cos^2\phi - \frac{r_1}{f}\sin\phi\cos\phi\right]}{-z - \frac{\sin^2\phi}{f} - \frac{r_1}{f^2}\sin\phi\cos\phi} \tag{24}$$

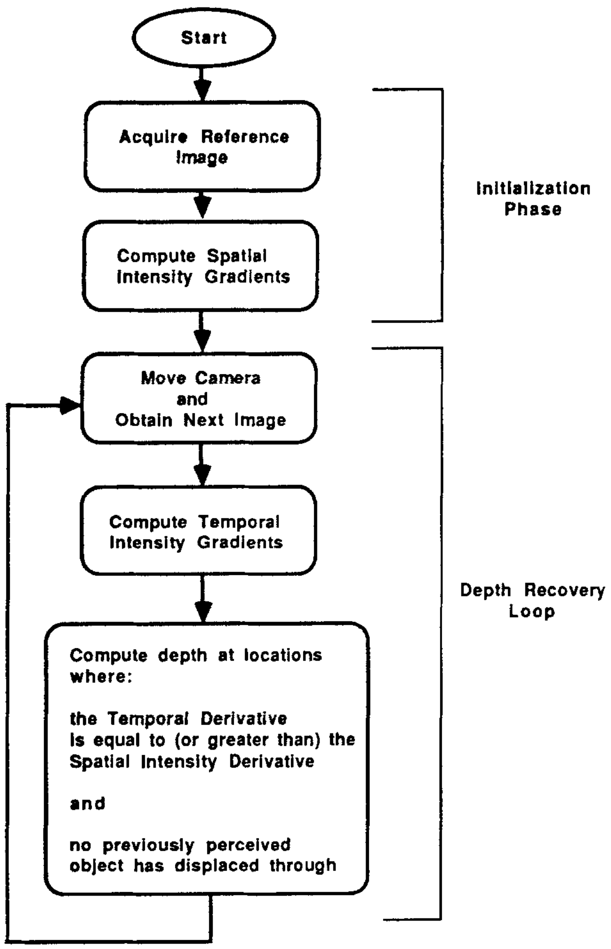
Start
Acquire Reference Image
Compute Spatial Intensity Gradients
Initialization Phase
Move Camera and Obtain Next Image
Compute Temporal Intensity Gradients
Compute depth at locations where:
the Temporal Derivative is equal to (or greater than) the Spatial Intensity Derivative
and
no previously perceived object has displaced through
Depth Recovery Loop

Figure 12. The IGA algorithm.

where δ is the distance the object displaced, z is the distance to the object, dz is the distance the camera has moved, r_1 is the distance from the center of the image to the location in the image where the object was first perceived, and ϕ is the angle between the optical axis of the camera and the direction of motion. Given δ for each previously perceived object on this particular displacement vector, it is possible to isolate those points due to new objects and compute depths directly using equation (21). (Note that ϕ, f, and dz are known constants, so the values of the transcendental functions need not be calculated and much of the computation can be replaced by table look-up.)

4 Potential Sources of Error

As with any vision algorithm, certain assumptions are made about the nature of the environment and equipment used for implementing IGA. In addition to the usual assumptions of stationary objects and constant illumination (Lucas 1985; Lucas and Kanade 1985; Herman and Kanade 1986; Negahdaripour and Horn 1986, 1987; Bolles et al. 1987; Jain et al. 1987; Tsukiyama and Huang 1987; Xu et al. 1987), IGA assumes that the camera's angle of orientation (ϕ) and displacement (dz) can be measured precisely (this is not unreasonable given the accuracy of optical-quality equipment).

The most significant potential sources of error in the IGA algorithm are artifacts of the geometry of the conventional imaging array. Since conventional imaging arrays are arranged in a row-column matrix format, they do not lend themselves well to the analysis of arbitrary displacement vectors (other than those with orientations that are multiples of $\pi/2$). Therefore, the determination of δ (disparity) may be off by as much as $\sqrt{2} - 1$ (a location exactly one pixel away along the displacement vector may lie in the receptor field of the current pixel's diagonal neighbor, which is actually 1.414 pixels away). A worst-case error potential of over 40% is certainly not acceptable in most situations; however, this does not at all detract from the potential of this algorithm when implemented in VLSI, where the imaging array could be mapped in polar form. Also, the potential for this large error occurs only at small numbers of points in a conventional image. Experimental results have shown that, when implemented using conventional imaging systems, the algorithm produces accurate results (see Section 5).

Since not all receptors have exactly the same spectral sensitivity, and random noise is a problem with conventional imaging systems, the assumption that depth can be recovered at all locations with nonzero spatial intensity gradients does not hold. For implementation purposes, it must be assumed that the gray-levels (0–255) returned by the imaging device are only accurate to five or six bits; that is, only locations with spatial intensity gradients greater than eight or so gray-levels are considered.

Because it is not practical to acquire images continuously, the sequence of images given to the IGA algorithm must represent a sparse sampling of the images seen by the observer as the camera undergoes translation. This introduces another potential source of uncertainty in the depth determination process. Consider the following situation (see Machine 1 in the left-hand side of the scene in Axial Sequence 2, Figure 18): an object is located 84 inches from the camera when the reference image is obtained. Suppose that this object appears 50 pixels from the center of the image, and that the camera is moved in increments of one inch. Obtaining two images ($dz = 1.0$ and $dz = 2.0$) using motion along the optical axis of the camera ($\phi = 0$), the IGA algorithm is used to determine the location of this object. The object will not be perceived in the first image because:

$$84 < 50 \times dz \tag{25}$$

The object is perceived in the second image, but the depth determined for the object is:

$$z = 50 \times 2.0 = 100 \tag{26}$$

which is incorrect. This error can be reduced by acquiring images at smaller increments (experimentally, acquiring images every 0.2–0.5 inches, seems to work well), but never eliminated because of the discrete nature of the problem. The error in the depth calculation resulting from an inaccurate estimate of dz propagates linearly through the computation (i.e., an error of 2% in dz will result in an error of 2% in z).

5 Experimental Results

IGA has been applied to several image sequences (of varying complexity) with a great deal of success. Results are presented for *five* different image sequences, three obtained using camera motion along its optical axis and two obtained using camera motion perpendicular to the optical axis (lateral motion). In presenting these results it is our intention to demonstrate the robustness of this technique. The execution times for all five sequences are also given.

It should be noted that the output given is, essentially, raw data. A simple smoothing filter has been applied to eliminate spurious errors due to sensor noise, however. The next logical step in the processing of this data would be to apply some sort of surface interpolation algorithm, or use this data to update a three-dimensional world model.

5.1 Axial Camera Motion

Figure 13A–E shows a relatively simple laboratory scene. In this particular example, 256 × 256 images have been acquired with camera displacements of 0.25, 0.5, 1.0, and 2.0 inches. The camera has been moved along its optical axis (Axial Motion Stereo Camera model), putting the FOE at image point (128,128). Figure 14 shows the map of actual object locations. In this and all subsequent examples, depth maps will be encoded with the gray-level corresponding to the depth (i.e., intensity 100 indicates an object 100 inches from the observer). White (intensity 255) is used to indicate unknown regions—those regions for which no information is recovered.

Table 1. Depth Values Returned for Axial Sequence 1

Actual vs. Perceived Depths Axial Sequence 1		
Object	Actual Depth	Perceived Depth
Desk lamp	30	29
Block	62	83
Cables	52	58
Box	44	57

Figure 15 shows the results of using only the reference image and the first images in the sequence ($dz = 0.25$); note that parts of the desk lamp are now being perceived ($z = 30$). This is as expected, as the edges of the desk lamp appear 100–140 pixels from the FOE in the reference image. Figure 16 shows the results of using all five images in the sequence; note that information is provided for all the objects in the field of view. Figure 17 shows a worst-case two-dimensional projection of the depth values returned by IGA for this sequence. Table 1 shows the depth values returned for some objects in the field of view. Note how these compare with the actual values.

Figure 18A and B shows the first and last images from a sequence taken of a somewhat more complicated laboratory scene. The orientation and relative locations of the objects in the field of view make this sequence similar to one that might be obtained by a mobile robot. Again, 128 × 128 images have been acquired using the axial motion stereo camera model. This time camera displacements of 0.5, 1.0, 2.0, 4.0, and 8.0 inches have been used. Figure 19 shows the map of actual object locations. The output of the IGA algorithm (after being applied to all six images) is shown in Figure 20. Again, a worst case two-dimensional projection of the depth values is shown in Figure 21. Table 2 compares the computed values to the actual object locations. We see that, again, the results compare quite favorably. Note that the error in depth for Machine 1 is an artifact of sampling too sparsely along the axis of motion (see previous section).

Table 2. Depth Values Returned for Axial Sequence 2

Actual vs. Perceived Depths Axial Sequence 2		
Object	Actual Depth	Perceived Depth
Tripod leg	40	47
Table	180	181
Machine 2	156	168
Machine 1	84	103

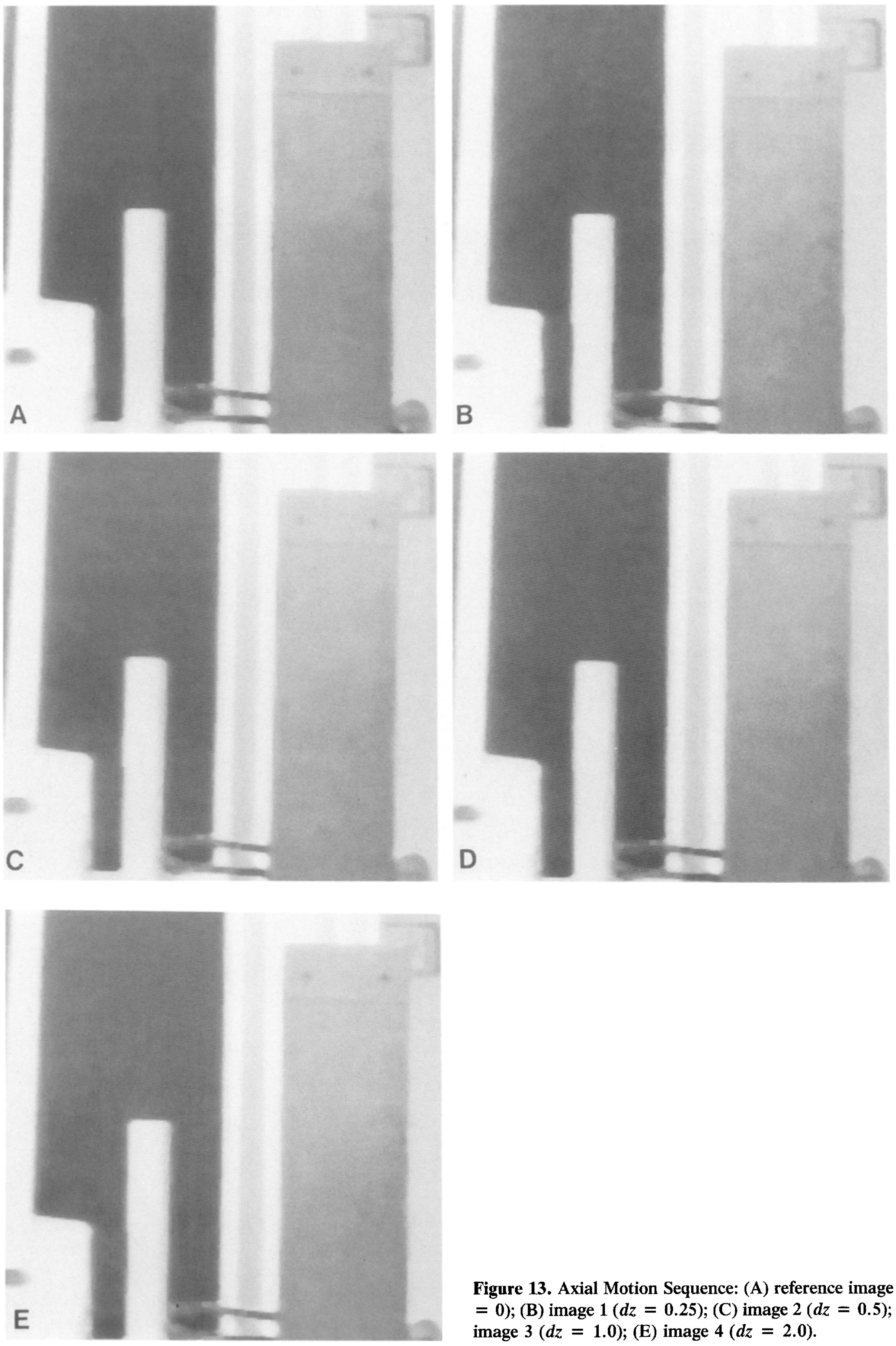

Figure 13. Axial Motion Sequence: (A) reference image ($dz = 0$); (B) image 1 ($dz = 0.25$); (C) image 2 ($dz = 0.5$); (D) image 3 ($dz = 1.0$); (E) image 4 ($dz = 2.0$).

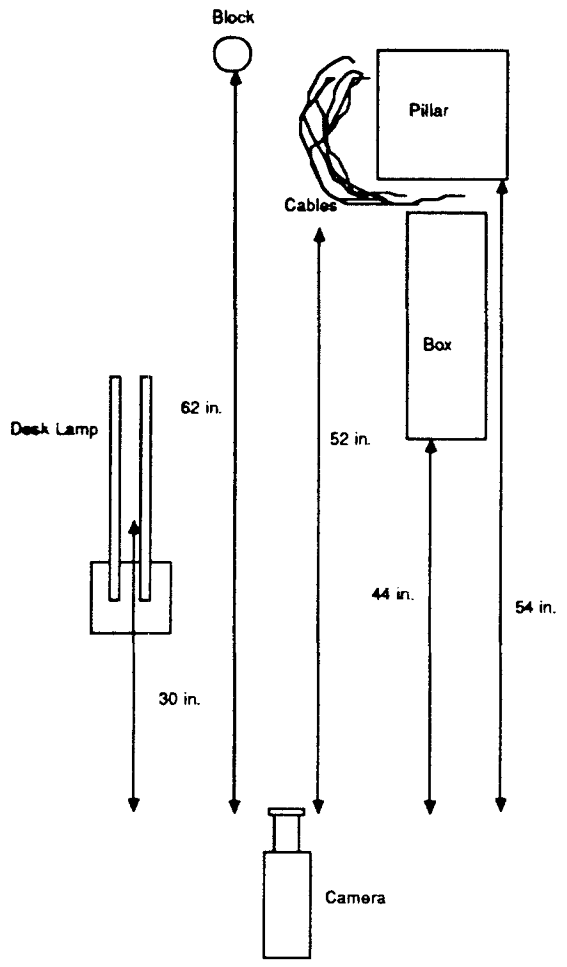

Figure 14. Map of object locations.

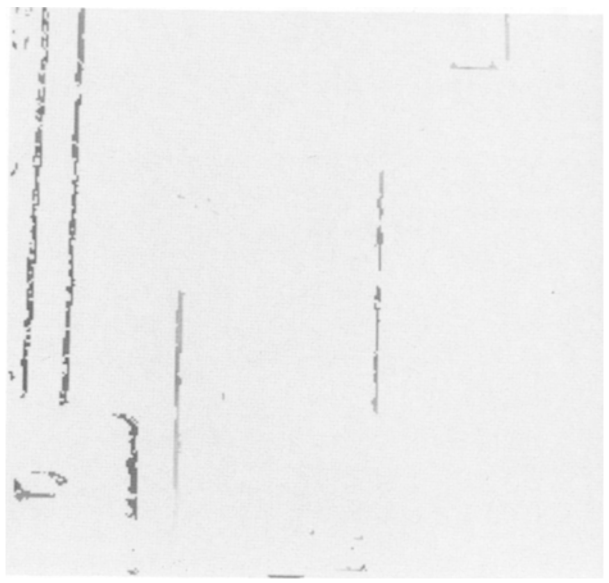

Figure 16. Depth Map obtained using all five images from the sequence.

Figure 22A–F shows another complicated laboratory scene. This time, 240 × 240 images have been acquired (again, using the axial motion stereo camera model) with a dense sampling of camera displacements (0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, and 4.0 inches). Figure 23 shows the map of actual object locations. The output of the IGA algorithm (af-

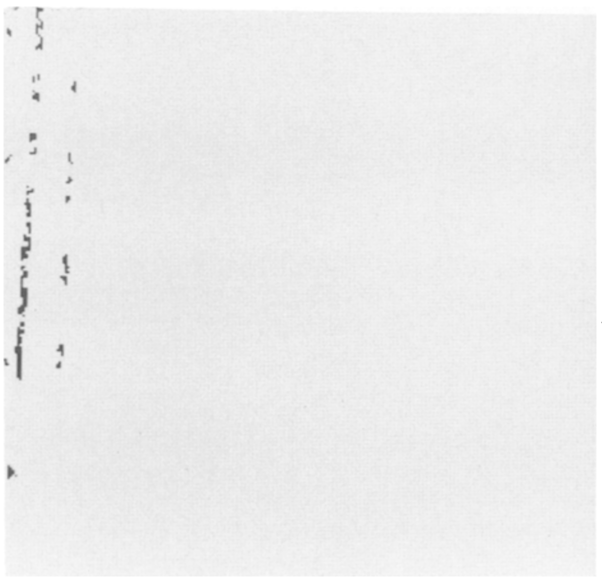

Figure 15. Depth Map obtained using the reference image and image 1 ($dz = 0.25$).

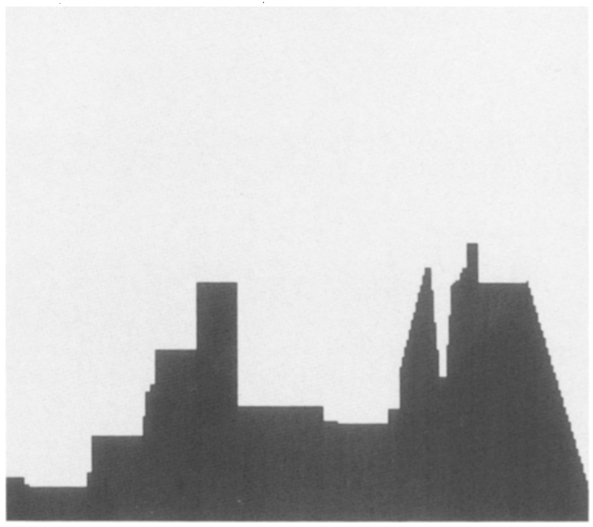

Figure 17. Worst-case two-dimensional projection of range data on the $x - z$ plane. Black indicates void space, white indicates object or unknown. The z-axis is oriented vertically, with up corresponding to increasing z.

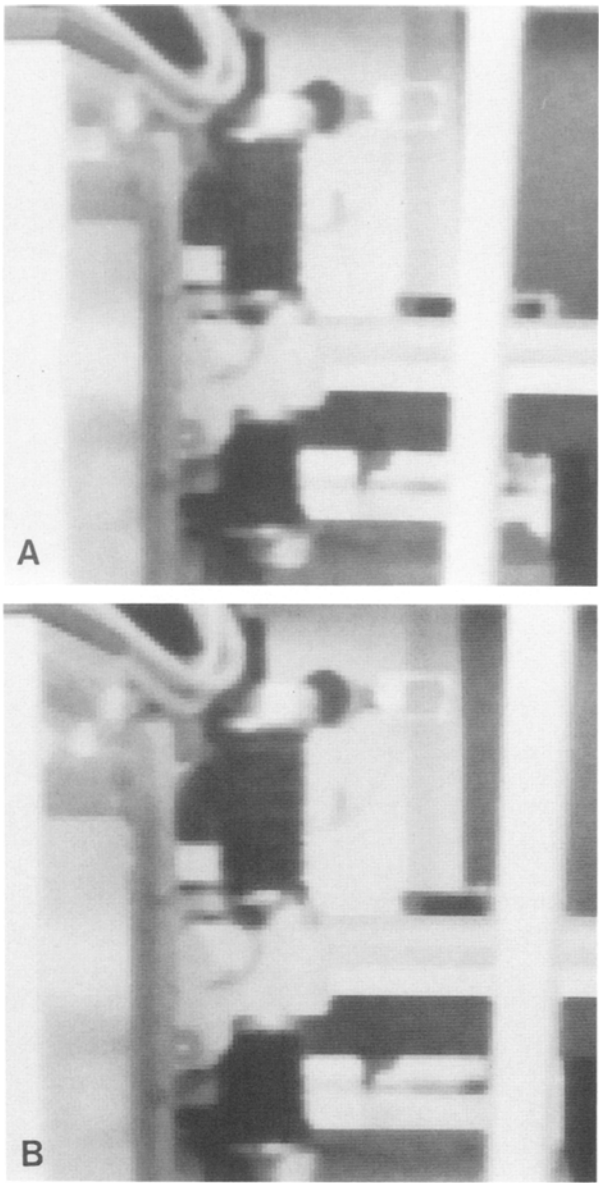

Figure 18. Axial Motion Sequence 2: (A) reference image (dz = 0); (B) image 5 (dz = 8.0).

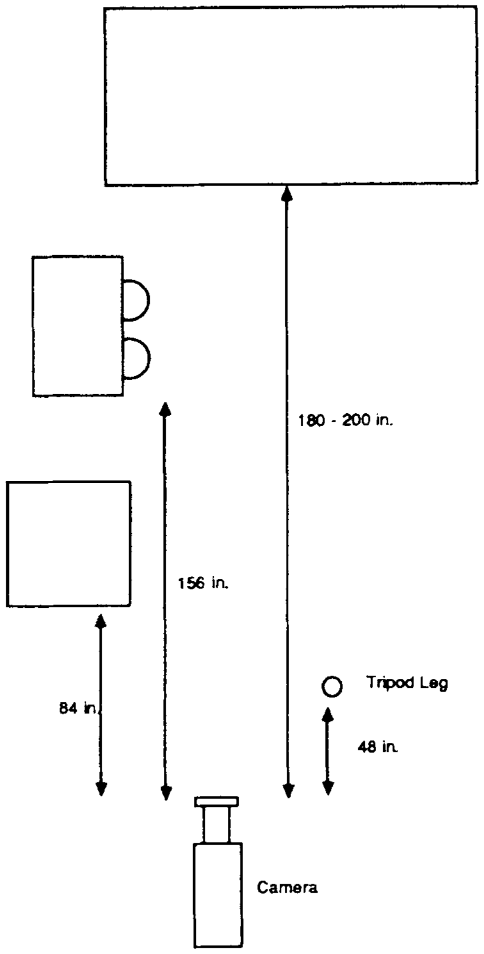

Figure 19. Map of object locations.

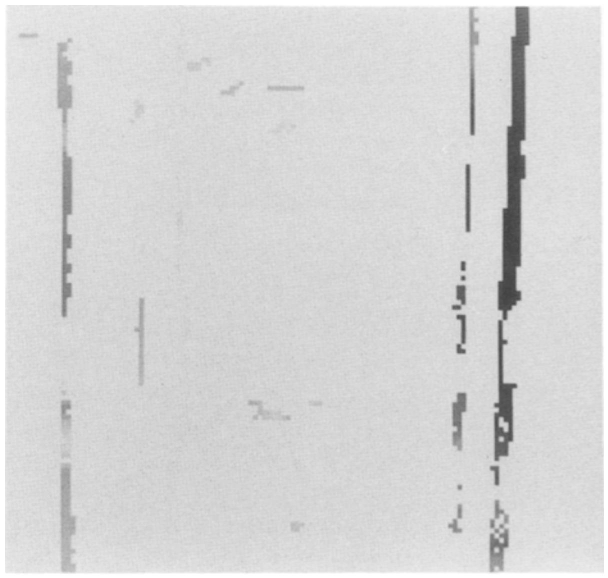

Figure 20. Depth map obtained using all six images from the sequence.

ter being applied to all nine images) is shown in Figure 24. Again, a worst case two-dimensional projection of the depth values is shown in Figure 25. Table 3 shows the depth values computed for the objects in the field of view. Note that no values are returned for the TV in the background, because it was too close to the FOE to be perceived with a dz of 4.0 inches.

5.2 Lateral Camera Motion

Figure 26A and B shows the first and last images taken from a sequence obtained using lateral cam-

Table 3. Depth Values Returned for Axial Sequence 3

Actual vs. Perceived Depths Axial Sequence 3		
Object	Actual Depth	Perceived Depth
Cabinet	60	62
Machine	120	125
Pallet	100	144

era motion. In this example, six 240 × 240 images were used, with camera displacements of 0.1, 0.2, 0.3, 0.4, and 0.5 inches. For these experiments, the lens parameters were not known, so depth values obtained are accurate only to a scale factor.

Figure 27 shows a map of object locations, and Figure 28 shows the depth map returned by the IGA algorithm. Figure 29 shows a worst-case two-dimensional projection of these depth values. Again, note the accuracy of this technique (Table 4).

Figure 30A and B shows the first and last images taken from a much more complicated sequence, obtained using lateral camera motion. Note that this is the same laboratory scene as was used for the third axial motion sequence. In this example, six 240 × 240 images were used, with camera displacements of 0.1, 0.2, 0.3, 0.4, and 0.5 inches. The actual locations of the objects are shown in Figure 23. Figure 31 shows the depth map returned by the IGA algorithm, and Figure 32 shows a two-dimensional projection of these depth values. Note that, even in this

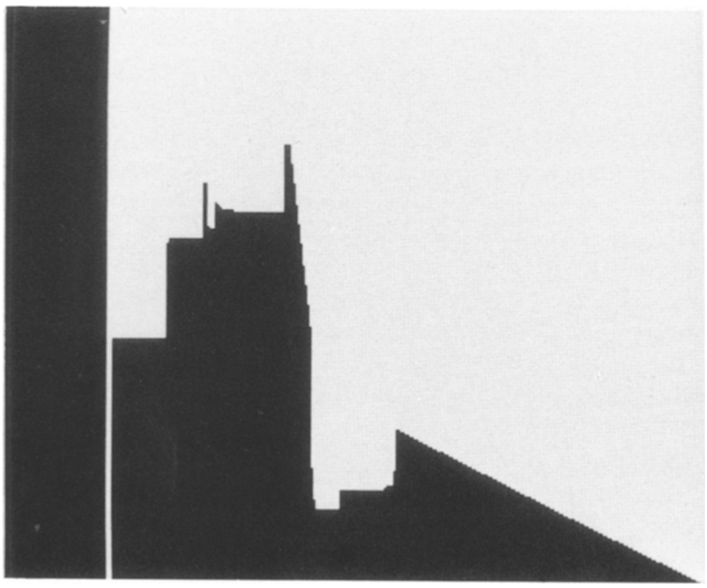
Figure 21. Worst-case two-dimensional projection of range data on the $x - z$ plane. Black indicates void space, white indicates object or unknown. The z-axis is oriented vertically, with up corresponding to increasing z.

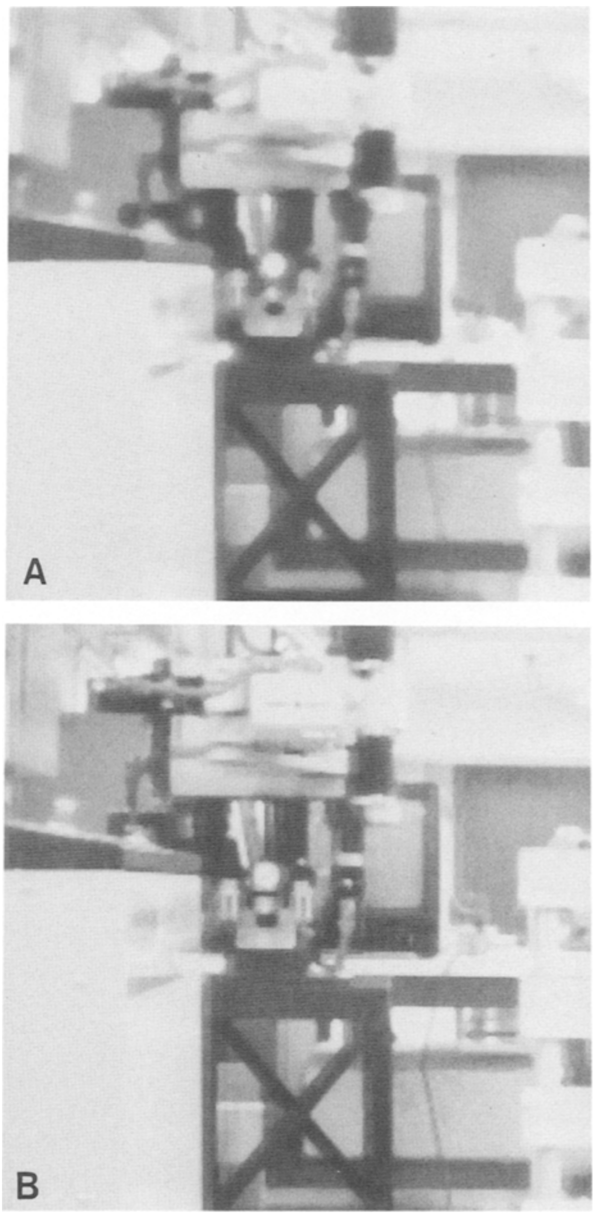

Figure 22. Axial Motion Sequence 3: (A) reference image ($dz = 0$); (B) image 8 ($dz = 4.0$).

Table 4. Depth Values Returned for Lateral Sequence 1

Actual vs. Relative Depths Lateral Sequence 1		
Object	Actual Depth	Relative Depth
Poster 1	100	44
Chair	170	120
Pallet	100	144
Cabinet	230	200

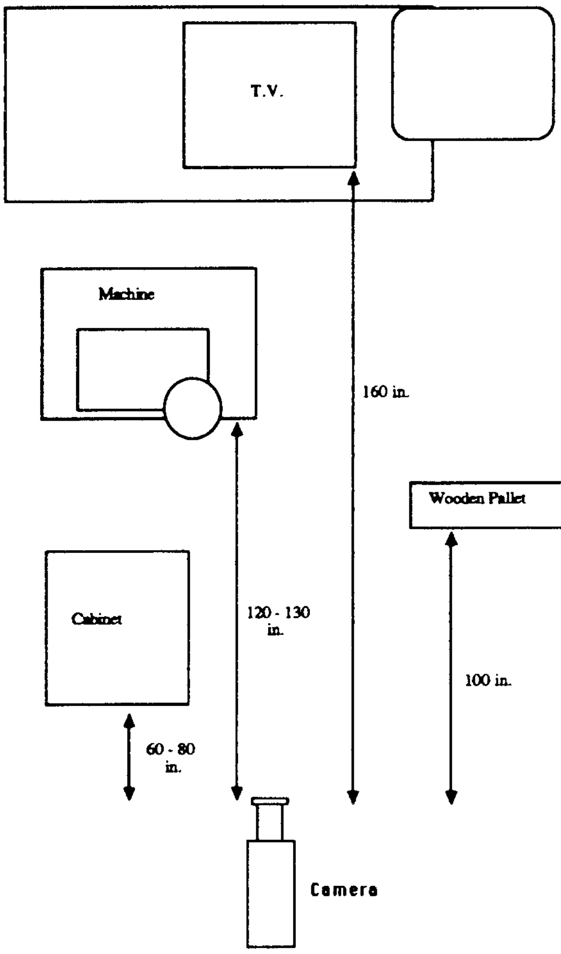

Figure 23. Map of object locations.

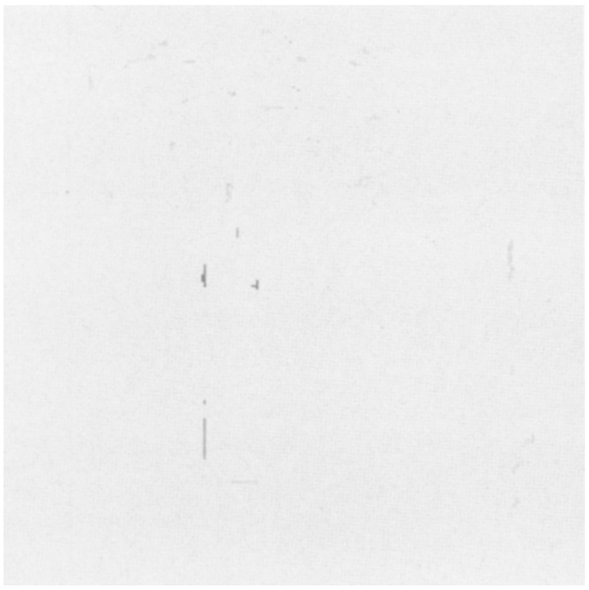

Figure 24. Depth map using all nine images from the sequence.

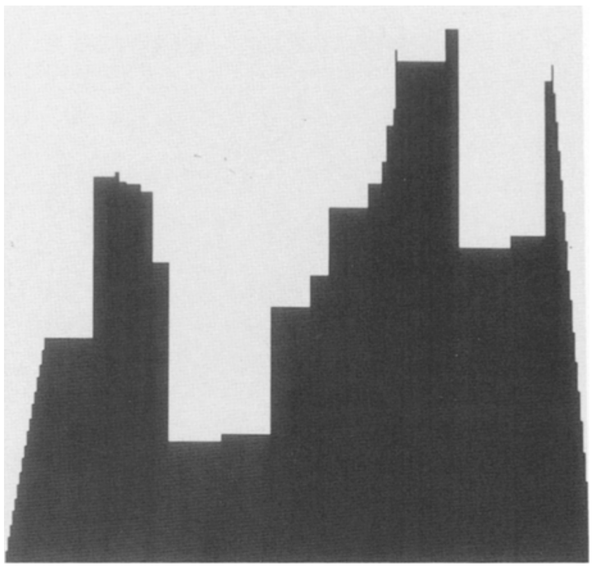

Figure 25. Worst-case two-dimensional projection of range data on the $x - z$ plane. Black indicates void space, white indicates object or unknown. The z-axis is oriented vertically, with up corresponding to increasing z.

complicated scene, qualitatively the results compare quite favorably with the actual values, although the quantitative results are not quite as good (Table 5).

5.3 Execution Times

Probably the most appealing property of the IGA algorithm is its speed. In the IGA paradigm, depth is recovered by doing a subtraction operation (to find the temporal intensity gradient), comparing the result to the spatial intensity gradient (which can be stored in a look-up table), and then, if indicated, performing the depth computation. It is not difficult to see that the computational requirements of IGA are minimal. Table 6 shows the execution times for the IGA algorithm on the five test sequences. Timing experiments were performed using code that is not optimized running on an Apollo DN4000 workstation that is part of a very large computer network. The times given are the total execution time

Table 5. Depth Values Returned for Lateral Sequence 2

Actual vs. Relative Depths Lateral Sequence 2		
Object	Actual Depth	Relative Depth
Cabinet	70	50
Machine	125	100
Pallet	100	160

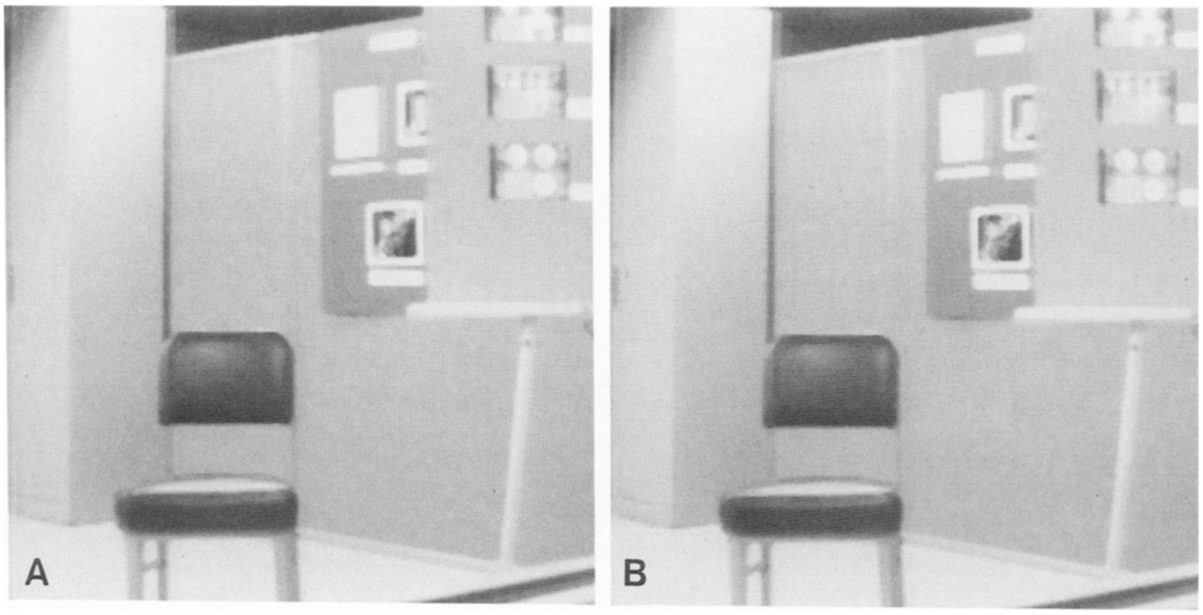

Figure 26. Lateral Motion Sequence 1: (A) reference image ($dz = 0$); (B) image 5 ($dz = 0.5$).

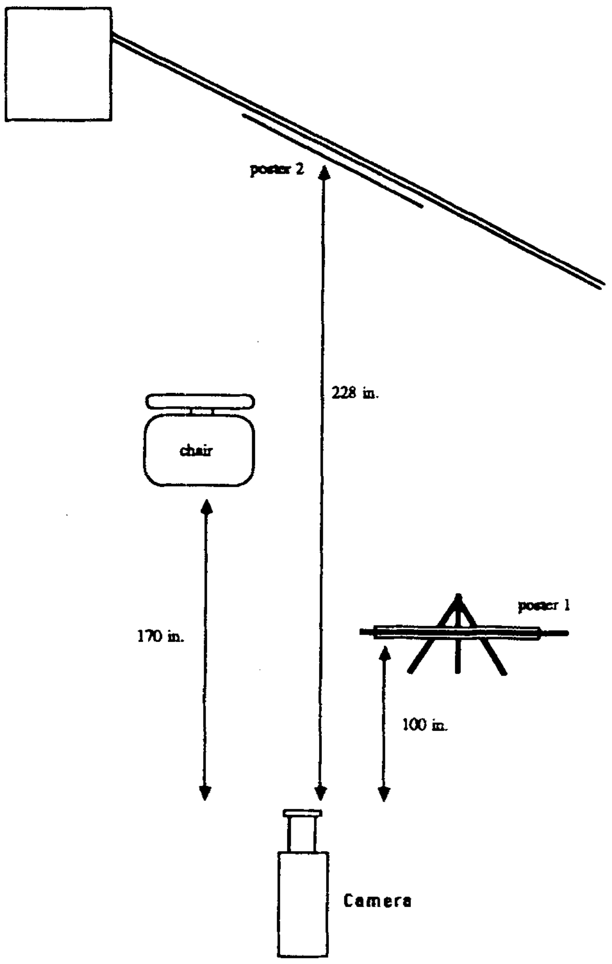

Figure 27. Map of object locations.

for the entire program, including the initialization of the look-up tables and all the file I/O, and the time spent in the depth recovery loop (including reading in the images). Nonetheless, these times are still quite impressive. Because of the local nature of all the computations, the IGA algorithm lends itself well to parallelization, and implications of implementing this in VLSI are quite promising.

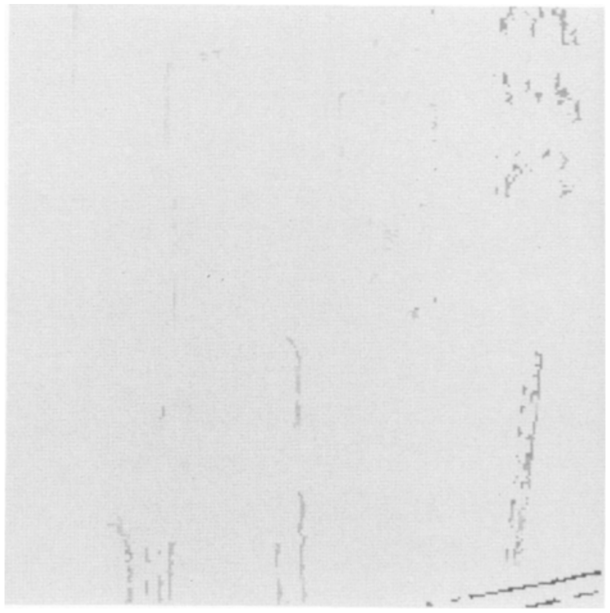

Figure 28. Depth map obtained using all nine images from the sequence.

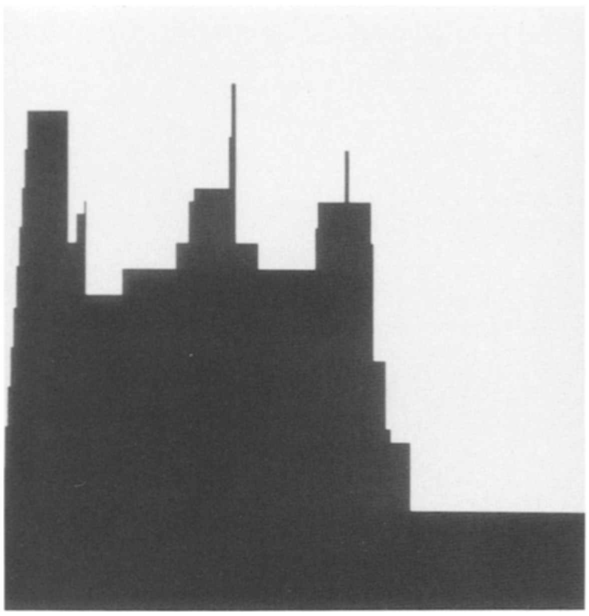
Figure 29. Worst-case two-dimensional projection of range data on the $x - z$ plane. Black indicates void space, white indicates object or unknown. The z-axis is oriented vertically, with up corresponding to increasing z.

6 Conclusions

Because the IGA technique places little burden on computational resources, the IGA algorithm seems ideally suited for applications such as autonomous vehicles (where computational resources are limited). This is a significant advantage over other techniques. It seems that by the time depth values are obtained from a single viewpoint using conventional techniques, the IGA algorithm may be able to give you depth information from *several views*. Even if one trades a little in accuracy, it seems that it would be quite advantageous to obtain the additional information. This is especially true when one considers the recent advancements in the integration of information from various sources (several

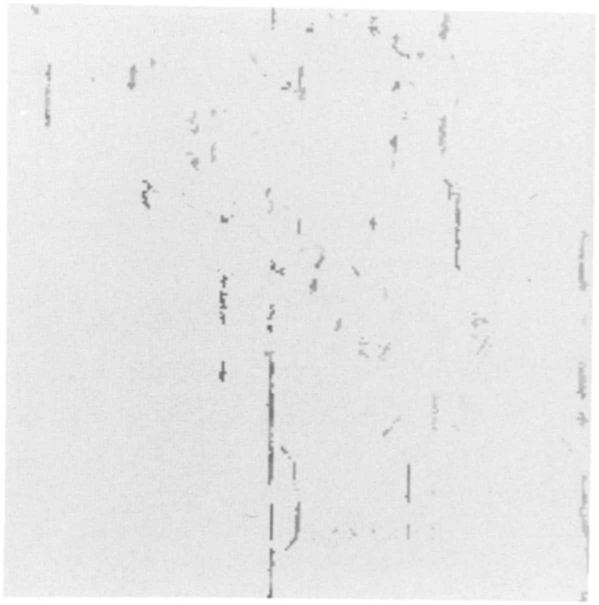
Figure 31. Depth map obtained using all nine images from the sequence.

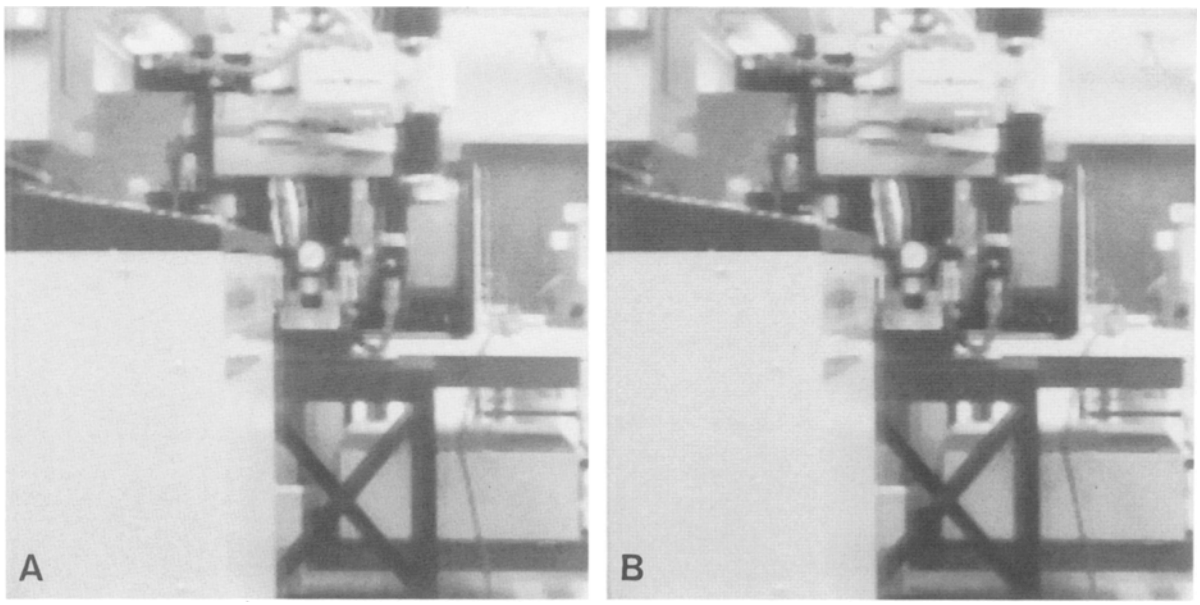

Figure 30. Lateral Motion Sequence 2: (A) reference image ($dz = 0$); (B) image 5 ($dz = 0.5$).

Table 6. Execution Times

Sequence	Image Size	No. of Image Pairs Processed	Total Execution Time (sec)	Time to Recover Depth Values (sec)
Axial sequence 1	128 × 128	4	5.8	3.7
Axial sequence 1	256 × 256	4	20.9	14.3
Axial sequence 2	128 × 128	5	9.4	7.1
Axial sequence 2	256 × 256	5	25.4	19.2
Axial sequence 3	120 × 120	8	5.8	4.1
Axial sequence 3	240 × 240	8	23.4	17.6
Lateral sequence 1	120 × 120	5	3.5	2.6
Lateral sequence 1	240 × 240	5	13.1	10.6
Lateral sequence 2	120 × 120	5	3.9	2.9
Lateral sequence 1	240 × 240	5	13.8	11.3

different views) using techniques such as the Kalman filter (Mattheis et al. 1987).

We have shown that existing techniques, although conceptually appealing and mathematically elegant, are far too burdensome computationally for many applications. This computational burden is largely the result of the need to explicitly solve the correspondence problem. By approaching the problem from more of an engineering viewpoint, we have shown that it is possible to recover depth values from gray-level imagery *without* explicit solution of the correspondence problem, using simple IGA.

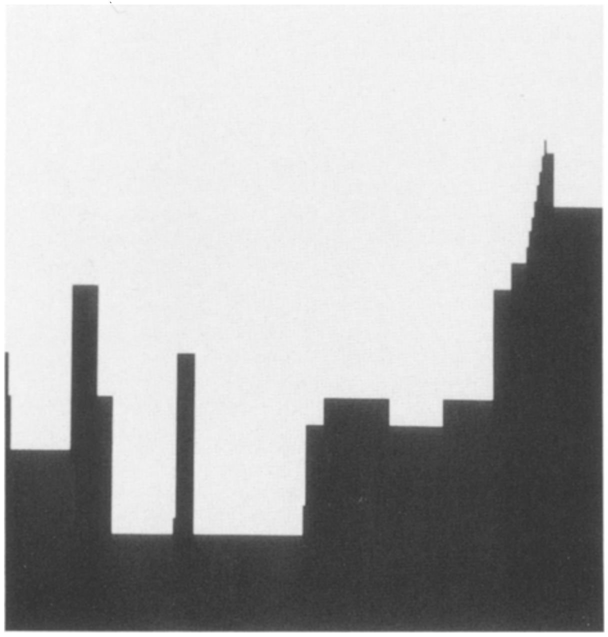

Figure 32. Worst-case two-dimensional projection of range data on the $x - z$ plane. Black indicates void space, white indicates object or unknown. The z-axis is oriented vertically, with up corresponding to increasing z.

Acknowledgment. This work was sponsored in part by Department of Energy Grant DE-FG02-86NE37969.

References

Alvertos N, Brzakovic D, Gonzalez RC (1988) Stereo camera modeling and image matching for 3-D machine vision. Proceedings of 1988 Conference on Computer Vision and Pattern Recognition, June 5–9, Ann Arbor, MI

Baker HH, Bolles RC (1988) Generalizing epipolar-plane image analysis on the spatiotemporal surface. Proceedings of 1988 Conference on Computer Vision and Pattern Recognition, June 5–9, Ann Arbor, MI

Barnard ST, Fischler MA (1982) Computational stereo. Computing Surveys (4), 553–572

Barnard ST, Thompson W (1980) Disparity analysis of images. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2 (4), 333–340

Bolles RC, Baker HH, Marimont DH (1987) Epipolar-plane image analysis: an approach to determining structure from motion. International Journal of Computer Vision 1:7–55

Boyer KL, Kak AC (1988) Structural stereopsis for 3-D vision. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-10:144–166

Clement RA (1987) Line correspondence in binocular vision. Perception 16:193–199

Collet TS, Harkness LIK (1982) Depth vision in animals. In: Ingle DJ, Goodale MA, Mansfield RJW (eds) Analysis of visual behavior. MIT Press, Cambridge, Massachusetts

Goldstein EB (1984) Sensation and perception. Wadsworth Publishing Co., Belmont, California

Grimson WEL (1981) From images to surfaces: a computational study of the human early visual system. M.I.T. Press, Cambridge, Massachusetts

Griswold NC, Yeh CP (1988) A new stereo vision model based upon the binocular fusion concept. Computer Vision, Graphics and Image Processing 41:153–171

Herman M, Kanade T (1986) Incremental reconstruction of 3D scenes from multiple, complex images. Artificial intelligence 30:289–341

Horn BKP (1986) Robot vision. MIT Press, Cambridge, Massachusetts

Itoh H, Miyauchi A, Ozawa S (1984) Distance measuring method using only simple vision constrained for moving robots. In: Proceedings of the Seventh International Conference on Pattern Recognition. Vol. 1. Montreal, pp 192–195

Jain R, Bartlett SL, O'Brien N (1987) Motion stereo using ego-motion complex logarithmic mapping. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9:356–369

Kanade T, Thorpe C (1986) CMU strategic computing vision project report: 1984–1985. CMU Technical Report, CMU-RI-TR-86-2

Lee DN (1976) A theory of visual control of braking based on information about time-to-collision. Perception 5:437–459

Lee DN (1980) The optic flow field: the foundation of vision. Philosophical Transactions. Royal Society of London, B290:169–179

Lucas BD (1985) Generalized image matching by the method of differences. CMU Technical Report, CMU-CS-85-160

Lucas BD, Kanade T (1985) Optical navigation by the method of difference. In: Proceedings of IJCAI 1985. pp 981–983

Mattheis L, Szeliski R, Kanade T (1987) Kalman filter-based algorithms for estimating depth from image sequences. CMU Technical Report, CMU-CS-87-185

Mayhew JEW, Frisby JP (1981) Psychophysical and computational studies towards a theory of human stereopsis. Artificial Intelligence 17:349–385

Moravec HP (1981) Robot rover visual navigation. UMI Research Press, Ann Arbor, Michigan

Negahdaripour S, Horn B (1986) Direct passive navigation: analytical solution for planes. In: Proceeding of the IEEE Conference on Robotics and Automation. San Francisco, pp 1157–1163

Negahdaripour S, Horn B (1987) Direct passive navigation. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9 (1):168–176

Nevatia R (1984) Depth measurement by motion stereo. Computer Graphics and Image Processing 5:203–214

Nishihara HK (1984) Practical real-time imaging stereo matcher. Optical Engineering 23(5):536–545

O'Brien N, Jain R (1984) Axial motion stereo. In: Proceedings of Workshop Computer Vision. Annapolis, MD, pp 88–92

Prazdny K (1980) Egomotion and relative depth map from optical flow. Biological Cybernetics 36:87–102

Prazdny K (1985) Detection of binocular disparities. Biological Cybernetics 52:73–79

Tsukiyama T, Huang TS (1987) Motion stereo for navigation of autonomous vehicles in man-made environments. Pattern Recognition 20(1):105–113

Xu G, Tsuji S, Asada M (1987) A motion stereo method based on coarse-to-fine control strategy. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9:332–336

Yachida M, Kitamura Y, Kimachi M (1986) Trinocular vision: new approach for correspondence problem. In: Proceedings of the 8th ICPR. pp 1041–1044